**Webdriver IO :**

**What is WebdrierIO ?**

WebdriverIO lets you control a browser or a mobile application with just a few lines of code. Your test code will look simple and easy to read.

To Work with elements on a page has never been easier due to its synchronous nature. When fetching or looping over elements you can use just native JavaScript functions. With the $ and $$ functions which WebdriverIO provide.

**How to install webdriverIO**

**Step 1**:

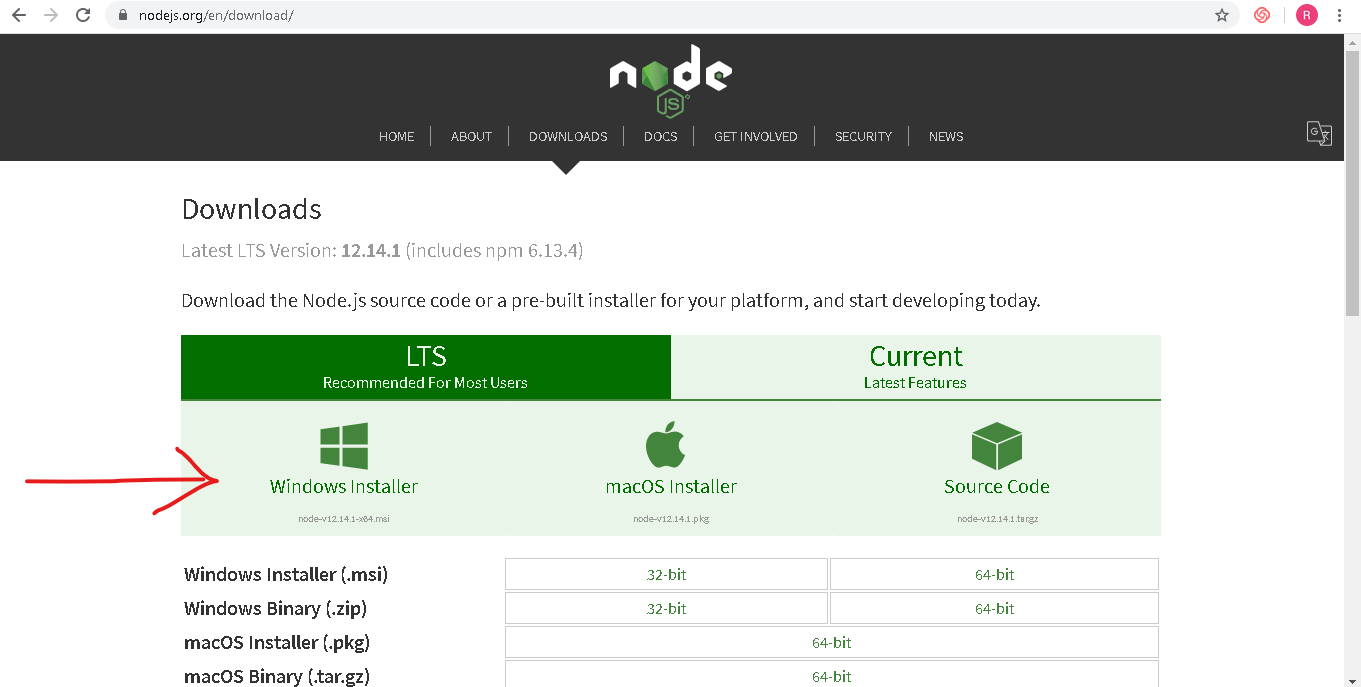
You’ll need Node.js installed (at least v8.11.2 or higher).

**Installation and Configuration**

**Download and install Node.js**

-Download Nodejs from [**https://nodejs.org/download/**](https://nodejs.org/download/)**. Download** installer .exe as per your OS/platform - save on your machine.

- Run the .exe, and finish the installation.



**Setting the Nodejs path in Environment system variable:**

Go to NodeJS installation at **C:\Program Files\nodejs** and copy this path and paste it at the end in Environment PATH system variable.

- Open a command prompt and type below command

**> node –version**

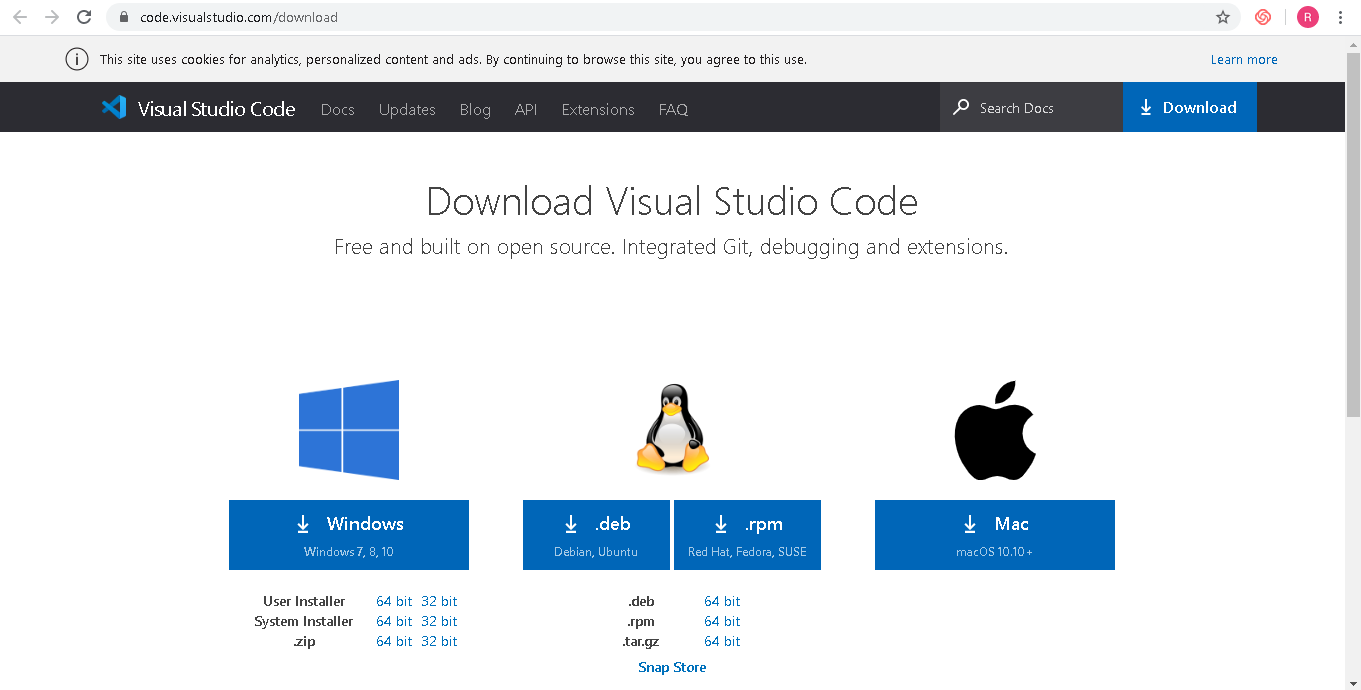
If it is showing version means it is installed properly

**Step 2 :**

**Download and install Visual Studio Code**

-Download Visual Studio Code from <https://code.visualstudio.com/download>

- Run the .exe, and finish the installation



**Step 3** :

mkdir webdriverio-test

cd webdriverio-test

npm init –y // initialize a new NPM project

**Step 4:**

Install Webdriver IO

npm I –save-dev @wdio/cli // Test runner file

**Step 5**

To do that, just run the configuration utility:

."/node\_modules/.bin/wdio" config –y //You have to generate configuration file

It will create a config file called  wdio.conf.js

**Step 6 :**

Now  we'll be ready to create your first spec file (or “test file”).

mkdir .\test\specs // Create a test folder

**Step 7 :**

Create a new file in it simple.js

Simply go to the folder and create a plain text file and name is as simple.js'

Now you will be able to write a script in this file

**Step 8 :**

To run this test file or js file you can use the command below

."/node\_modules/.bin/wdio" wdio.conf.js

./node\_modules/.bin/wdio wdio.conf.js (run from VS code)

**Now you are done with installation of webdriverIO and able to run first test**

* **Now we will learn about automation framework**

**Mocha** is a feature-rich JavaScript test framework. Mocha tests run serially,what tests you have and what tests you want to run and mocha will run your tests and report those that passed and those that failed. Mocha by itself provides a test running framework. You'll typically want to use an assertion library with it, like Chai. In test suites where the only libraries providing testing support are Mocha together with Chai.

Mocha: This is the JavaScript testing framework, but it cannot run as a standalone testing framework, it needs plugins or library to run as a testing framework. But it is easy to install. It is used at unit testing level.

npm install --global mocha // To install

mocha **–version //** To check version

7.1.1

First up we need to create a folder named test in our project. As default, Mocha looks for a test folder and then executes any test files that exist in that folder:

**How to create folder**

**mkdir test**

Once you have your test folder, create a js file.

**var** assert = require('assert');

**// We start by using the describe function in which our tests are created.**

describe('Maths calculation', **function**() {

**// The 'it' function is next to call, where we can name our test and then**

**// execute the code we want to run for our test.**

it('should return the correct value when two numbers are added', **function**() {

**// Here is the code for our test that is triggered by Mocha**

**var** result = 1 + 1;

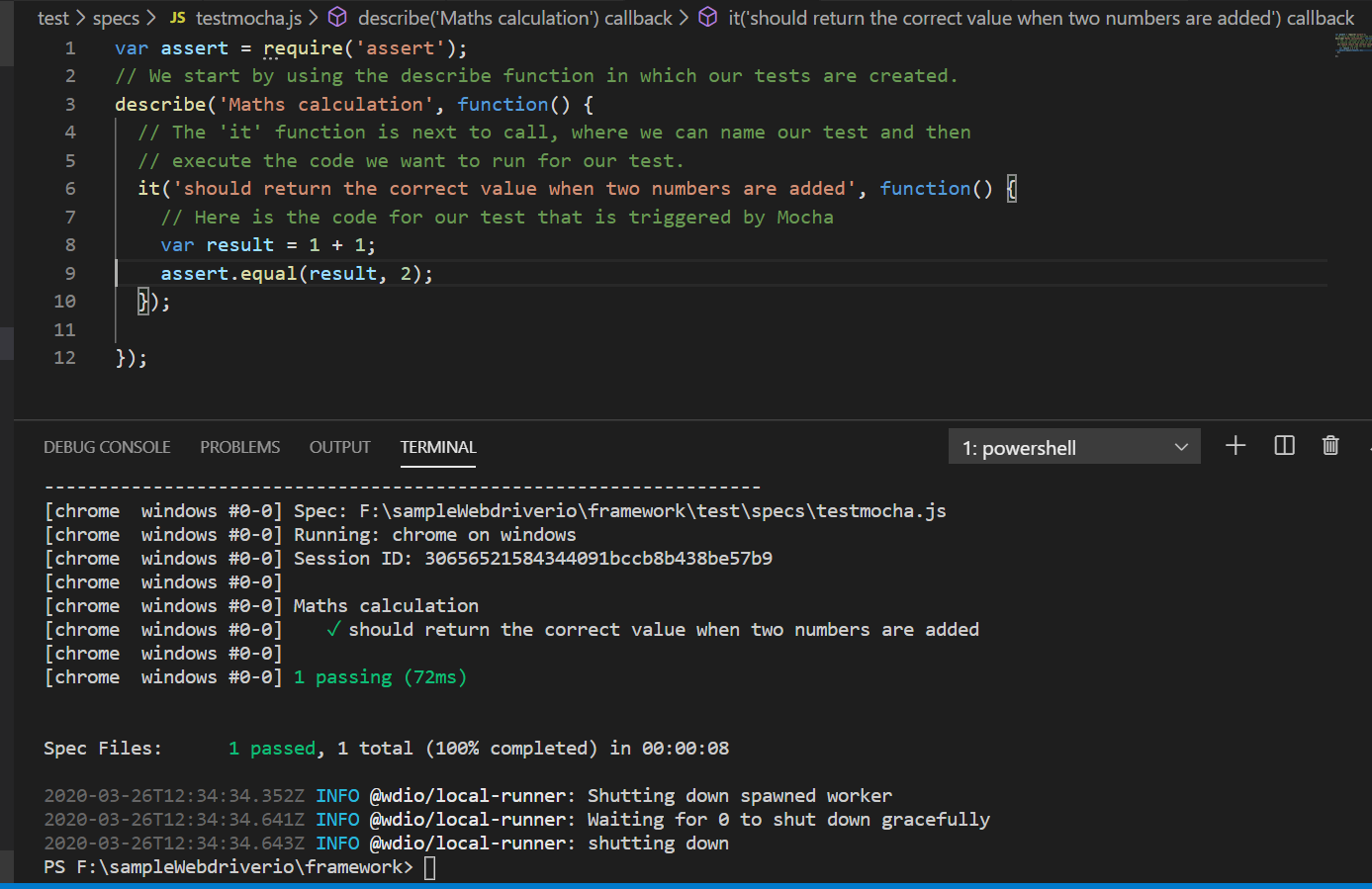
assert.equal(result, 2);

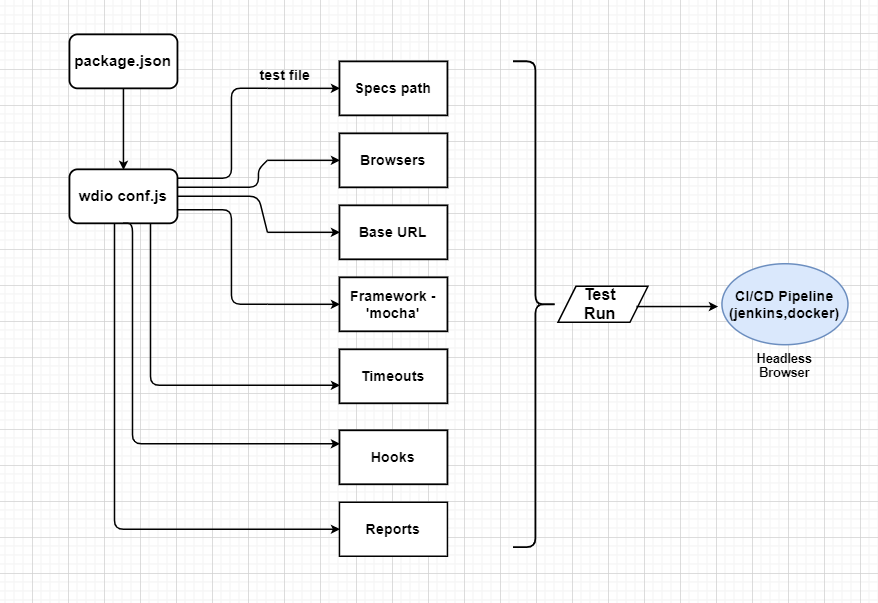
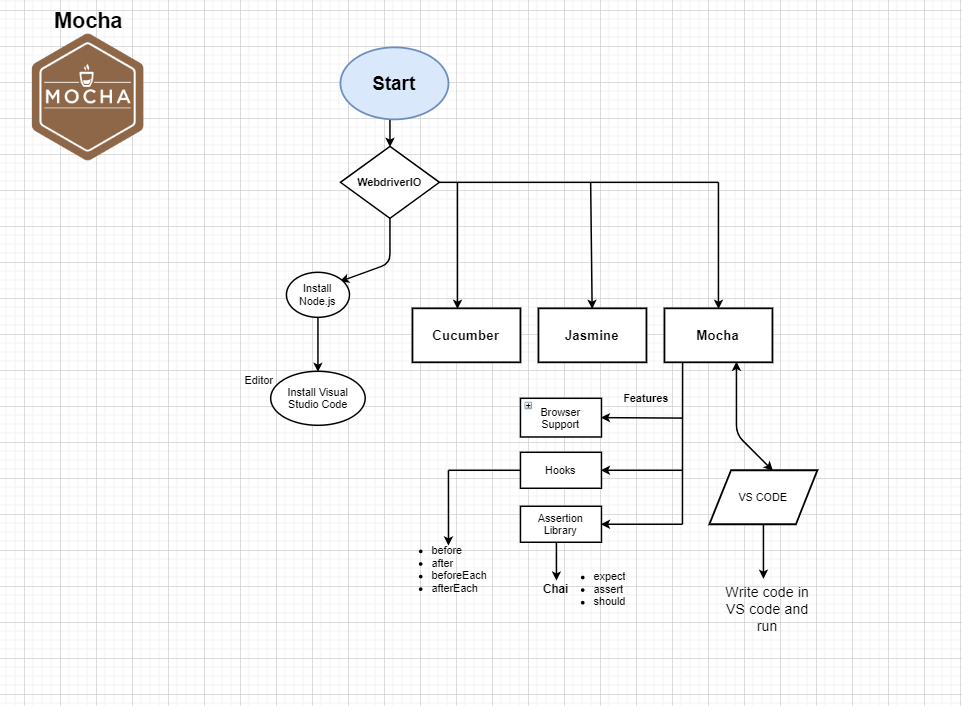
});

});

Now run this program

**./node\_modules/.bin/wdio wdio.conf.js**





**Explaination and Significance :**

* **Config**

: The Config folder contains the config file or any file like default.js in that file we can give our configuration like website name and some user ids and password or any small configurations like environment access.

* **Expected Result :**

Expected result contains the some assertion data like validations. After performing some validations we get the expected result

* **Lib:**

Lib contain some utility files/under development

* **Methods :**

We can create a methods in this folder those we want to use e.g LoginMethod , SignupMethod etc . All the universal methods we can keep in methods folder.

* **pageObjects :**

The goal of using page objects is to abstract any page information away from the actual tests. Ideally, you should store all selectors or specific instructions that are unique for a certain page in a page object, so that you still can run your test after you've completely redesigned your page.

Page Object contains all the modules . Every page class , methods and locators are present in Page Object folder

* **Reports :**

Allure test reports which is an HTML generated website with all necessary information to debug your test results and take a look on error screenshots.

* **Resourse :**

It is kind of data provider , in which we can put our sample data or web data.

* **screenshots :**

After running the scrips , the failed test cases screenshot will get add in this screenshot folder

* **test/specs** :

It is a normal directory in which we keep our all the test script. All the main and important scripts and we create in this

Test/specs folder.

* **Utils :**

Utils contain the random data or dummy data , or necessary library.What we need in our project that kind of dummy data we keep in Utils.

* **Docker file :**

This service is for use with [WebdriverIO](http://webdriver.io/) and it helps run functional tests using containerized applications. We have to install it separately to run.

To install docker : npm install wdio-docker-service --save-dev

* **Package.json :**

Package.json is a plain JSON(Java Script Object Notation) text file which contains all metadata information about Node JS Project or application. Every Node JS Package or Module should have this file at root directory to describe its metadata in plain JSON Object format.NPM (Node Package Manager) uses this package.json file information about Node JS Application. Package.json file contains a number of different directives or elements.

* **Package-lock.json :**

package-lock.json is automatically generated for any operations where npm modifies either the node\_modules tree, or package.json.

A new file was created automatically*.*Package-lock.json. If you open it, it looks sort of like the dependencies in package.json.

* **Wdio.conf.js :**

This file contains specs path , browsers , baseURL ,Framework, timeouts,hooks, etc . all the nessesery and config file are present in wdio.conf.js file. So when we run the script then it will find all the details from this file and will execute accordingly

--------------------------------------------------------------------

**How they communicate ?**

when start test - it checks for dependencies, configuration, then start as per the test suite mentioned in the config file –

then communicate as per page/object function call, libraries (random data), open browsers and execute test and display results on the screen and generate report/screenshots etc